A Distributed Community Detection Heuristic Approach for Large-scale Real Graphs

Guanglong Chi
Department of Human System Science
Tokyo Institute of Technology
Tokyo, Japan 152-8552
JST, CREST
Email: chi.g.aa@m.titech.ac.jp

Ken Wakita
Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
Tokyo, Japan 152-8552
JST, CREST
Email: wakita@is.titech.ac.jp

Hitoshi Sato
Global Scientific Information and Computing Center
Tokyo Institute of Technology
Tokyo, Japan 152-8552
JST, CREST
National Institute of Advanced Industrial Science and Technology
Email: hitoshi.sato@gsic.titech.ac.jp

***Abstract*—Louvain method [1] is considered as a representative community detection technique for greedily maximizing the modularity [2] of a partitioned network. In order to support large-scale graphs whose size exceeds the capacity of host memory on a single compute node, distributed methods, such as PMETIS-Louvain [3] and Hash-Louvain [4], etc., have been proposed. However, existing techniques employ vertex-oriented graph partitioning as a preprocess for fitting parts of a given graph on distributed compute nodes, and introduce time consuming overheads and waste memory consumption between distributed processes due to imbalances of edge numbers between partitioned graphs. In order to avoid such edge imbalance situation, we propose a distributed Louvain with edge-oriented graph partitioning that balances the number of edges between compute nodes and eliminates the performance overheads, while keeping good modularity results. Experimental results to Wiki graph show that our proposed technique achieves 23% reduction in maximum memory, and 6% in Pokec, and the proportion of max wait time to the community detection time is reduced from 6 to 0.6 in Wiki, and reduced from 6 to 0.7 in Pokec.**

I. Introduction

Community detection is one of the most essential analysis methodologies in complex network analysis. Louvain [1] is a representative greedy community detection method that achieves computational efficiency and scalability by means of agglomerative optimization of the *modularity measure* [2]. However, larger datasets do not fit in the primary memory of a single-node computers. For example, a Facebook acquaintances network that consists of 149 million user accounts and 31 billion connections can consume about 377GB in Louvain.

In order to handle large-scale social graphs, distributed methods such as PMETIS-Louvain [3] and Hash-Louvain [4] has been proposed. These approaches adopt a pre-processing technique that partitions a whole social graph into loosely connected subgraphs and assigns the subgraphs to compute nodes to allow distributed execution, where computation parts for modularity measure are independently conducted on distributed nodes and the partial results are infrequently exchanged for keeping good modularity results and minimizing communication overheads.

Both of previous proposals employ variations of *vertex-oriented partitioning* techniques that minimize the number of cross-edges between subgraphs. Vertex-oriented partitioning works good for small world networks but they suffer from serious edge-imbalancing for scale-free networks. Edge imbalance leads to memory imbalance, since the number of edges is commonly several times more than nodes in complex networks, and also leads to time imbalance since the computational complexity of Louvain method is roughly proportional to the number of edges. In the preliminary experiment, dividing the Wiki dataset into 8 parts by random vertex partitioning shows that 65% of all edges are accommodated in a single subgraph.

To address the above problem, we employ *edge-oriented partitioning* technique in the *partitioning stage* of a distributed Louvain method, in which we assign equal numbers of edges to distributed compute nodes. Experimental results show that maximum memory consumption is reduced by 23% for the Wiki dataset and 6% for the Pokec dataset respectively; and the proportion of maximum wait time to the community detection time is decreased from 6 to 0.6 for Wiki and from 6 to 0.7 in Pokec. The quality of the identified community structures is also equal to those results obtained from the original Louvain method.

II. Proposal

Modularity [2] is a metric for evaluating the quality of the community detection result and is defined as follows: $Q = \sum_{i=1}^{c}(e_{i,i} - a_i^2)$ where C means the set of all detected communities. $e_{i,j}$ represents the fraction of the edges between community i and community j. $a_i = \sum_j e_{i,j}$ represents the fraction of the edges that connect community i to others.

Louvain method initially treats each vertex as one community. Then it iteratively attempts to move a vertex to its neighboring community that locally maximizes the modularity measure. This iteration step performed once for all the vertex in the graph is called a *pass*. Passes are performed multiple times until the modularity measure converges. The whole set of passes is called a *level*. After each level, *contraction phase* replaces communities with vertices and merges edges

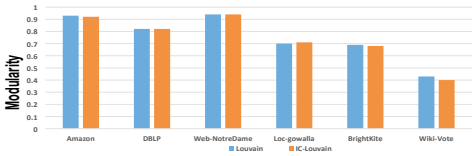
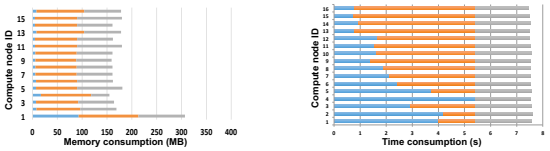


Fig. 1. Modularity comparison between Louvain (blue) and IC-Louvain (orange). The number of compute nodes is 4.



(a) PMETIS-Louvain: mem (b) PMETIS-Louvain: time

(c) IC-Louvain: mem (d) IC-Louvain: time

Fig. 2. Time and memory breakdown of Wiki

that connect members of a pair of communities with an edge connecting a pair of vertices resulting from contraction.

Implementations of distributed Louvain method, including our proposal, executes in three stages: (1) graph partitioning stage, (2) Louvain method computation stage, and (3) communication stage. As for graph partitioning stage, PMETIS-Louvain and Hash-Louvain method adopt vertex-oriented partitioning, while IC-Louvain (our proposal) adopts an edge-oriented partitioning called incremental cost assignment (IC) method [5]. Frequency of communication in three methods differ considerably. PMETIS-Louvain and IC-Louvain methods exchange large amount of information in *per-level* and *per-pass* manner, respectively. Hash-Louvain method, on the other hand, relies on frequent, asynchronous exchange of small pieces of information.

III. Experiment

Efficiency and quality of the results are equally important in community detection. As implementation of distributed Louvain methods employs a lazy approach with local optimization and infrequent communication for efficiency instead of global optimization, the results in distributed approaches may degrade in terms of modularity measure, mutual information, and similarity of community membership in a given network. We compare IC-Louvain method with the original Louvain method over 12 real social graph data obtained from the Stanford dataset collection [6], whose results exhibit good compromise as the original method as shown in Fig.1.

Next, we compare execution efficiency of IC-Louvain method with PMETIS-Louvain method using the larger datasets in the Stanford dataset, Wiki and Pokec, using 16 distributed compute nodes. Here we show the results of Wiki in Fig.2, in which Fig.2a and 2c show the breakdown of the memory consumption in each compute node, including graph data initialization, vertices to communicate (called *border*), and communication buffers. We see that IC-Louvain performs well-balanced memory consumption in graph data initialization and border vertices. Although IC-Louvain introduces frequent communications between distributed processes compared with PMETIS-Louvain, we also observe that IC-Louvain successfully achieves reduced maximum memory consumption; 23% in Wiki and 6% in Pokec. As such edge-oriented partitioning is efficient in balancing memory usage.

The improved edge imbalance with IC-Louvain method benefits it with better computational balancedness over the compute nodes, as observed in Fig.2d. Fig.2 compares the execution times for the first level of PMETIS-Louvain method (b) and the first pass of IC-Louvain method (d), respectively. Edge imbalance in the former method is the cause of significant imbalance of execution time; computation (blue) in nodes #1 through #5 puts other nodes that finish earlier to synchronization state (orange) for noticeable length of time. In contrast, such imbalance is resolved with IC-Louvain method. It should be noted that the aggregated synchronization cost amount almost the same as computational cost with PMETIS-Louvain method but in IC-Louvain method it is much smaller.

Although IC-Louvain method greatly improves computational efficiency over PMETIS-Louvain method, the former suffers from increased communication frequency. As communication frequency affects the quality of community detection, we can not simply reduce it. A better combination of graph partition technique and reduced communication scheme is yet to be searched.

IV. Conclusion

We focused on edge imbalance problem, and proposed a distributed Louvain with edge-oriented partitioning that can avoid it. IC-Louvain efficiently balanced memory usage and community detection time, however, it becomes slower than PMETIS-Louvain since the frequent communications. A method with edge imbalanced and per-level communication frequency is desirable, and it is our next goal.

Acknowledgment This work was partially supported by the JST-CREST Project.

References

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[2] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[3] C. Wickramaarachchi, M. Frincu, P. Small, and V. Prasanna, "Fast parallel algorithm for unfolding of communities in large graphs," in *Proc. High Performance Extreme Computing Conference (HPEC '14)*, 2014.

[4] X. Que, F. Checconi, F. Petrini, and J. A. Gunnels, "Scalable community detection with the louvain algorithm," in *Proc. Parallel and Distributed Processing Symposium (IPDPS '15)*. IEEE, 2015, pp. 28–37.

[5] F. Bourse, M. Lelarge, and M. Vojnovic, "Balanced graph edge partition," in *Proc. ACM SIGKDD*, 2014, pp. 1456–1465.

[6] S. N. A. Platform, https://snap.stanford.edu/data/.